№	Количество экспериметов	Тип тестовой модели	Средняя длина квазислучайной последовательности	Средняя длина псевдослучайной последовательности
1	100	Точечная	309	323
2	100	Точечная	437	500
3	100	Точечная	424	420
4	100	Узкополосная	35857	35789
5	100	Узкополосная	253456	252987
6	100	Узкополосная	9145	9378
7	100	Блочная	15167	16754
8	100	Блочная	16564	18143
9	100	Блочная	16572	17143

Таблица 1 – Сравнение квазислучайных и псевдослучайных последовательностей в зависимости от типа тестовой модели

Из приведенных выше данных видно, что длина квазислучайной тестовой последовательности на 7-10% меньше чем псевдослучайной. Наибольшую эффективность квазислучайные последовательности показали при тестировании блочной тестовой модели.

Список использованных источников:

1. Ярмолик, С.В. Квазислучайное тестирование вычислительных систем. / С.В. Ярмолик, В.Н. Ярмолик // Информатика. – 2013. - № 3 – С. 65-81.
2. Иванюк, А.А. Проектирование встраиваемых цифровых устройств и систем / А.А. Иванюк. – Минск : Бестпринт, 2012. – 338 с.
3. Ярмолик, В. Н. Адресные последовательности для многократного тестирования ОЗУ / В. Н. Ярмолик, С. В. Ярмолик // Информатика . - 2014. - N 2. - С. 124-136.

ИНСТРУМЕНТАЛЬНОЕ ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ РАЗРАБОТКИ МОДУЛЕЙ РАСШИРЕНИЯ ПРИЛОЖЕНИЙ MICROSOFT OFFICE

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Коваль Д. А.

Сурков К. А. – ассистент

Задача по автоматизации разработки модулей расширения приложений Microsoft Office (Excel, Word, PowerPoint) является актуальной в сфере разработки программного обеспечения, поскольку на сегодняшний день на рынке существует лишь одно по-настоящему мощное средство – Visual Studio Tools For Office. Целью разработки является программное средство CommonOffice, представляющее собой набор библиотек программирования, позволяющих автоматизировать процесс разработки модулей расширения приложений Microsoft Office на языках программирования C++ и C#, а так же интегрировать разработанные модули в один комбинированный модуль.

Принципы, положенные в основу поиска технического решения и проектирования программного средства, сформировались в условиях реальной необходимости такого программного средства, а именно: во время разработки коллективами программистов системы модулей расширения, были обнаружены следующие проблемы:

- децентрализованность прикладных интерфейсов программирования
- сложность во взаимодействии модулей
- дублирование кода
- дублирование данных
- проблемы с производительностью и потребляемой хостом оперативной памятью.

Приложения Microsoft Office (Excel, Word, PowerPoint) предоставляют возможности для своего расширения посредствам COM интерфейсов. Таким образом, с точки зрения хост приложений, разрабатываемое программное средство представляет собой COM плагин. Для интеграции в Office приложения в качестве COM плагина, необходимо разработать COM компонент реализующий IDTExtensibility2 COM интерфейс, экспортируемый Office приложениями, и зарегистрировать этот компонент в качестве COM плагина

для Office приложений в реестре операционной системы. После регистрации COM компонента в реестре, бинарный модуль, в котором он расположен (Shim), будет загружен в адресное пространство хост приложения. При успешной загрузки в память приложения, COM компонент из Shim модуля выполняет работу по развертыванию .NET инфраструктуры, а именно:

1. Создает в адресном пространстве хост приложения COM сервер CLR .
2. Создает домен приложений в CLR посредствам CLR COM интерфейсов.
3. Загружает в созданный домен приложений .NET сборку с модулем, представляющим собой управляемый плагин (CommonOffice).

После того, как управляемый модуль CommonOffice загружен, он начинает параллельную загрузку всех своих модулей.

Используя инфраструктуру CommonOffice платформы, можно легко создать плагин для Microsoft Office Excel, Word и PowerPoint. Для этого необходимо выполнить несколько действий, которые позволят CommonOffice найти, распознать, загрузить в память и управлять созданным модулем расширения:

1. Реализовать соответствующие интерфейсы из библиотек CommonOffice.
2. Создать XML описание модуля, в котором указать:
   2.1. Имя библиотеки, в которой находится реализация IModule интерфейса.
   2.2 Имя класса, реализующего IModule интерфейс.
   2.3 Признак, говорящий, надо ли изолировать данный модуль в отдельном домене.
3. Создать XML описание для Microsoft Office Ribbon UI (при необходимости).
4. Поместить библиотеку с реализацией IModule и XML описание модуля в определенный каталог на жестком диске.

Общий механизм инициализации CommonOffice в Office приложения отображен на рисунке 1.

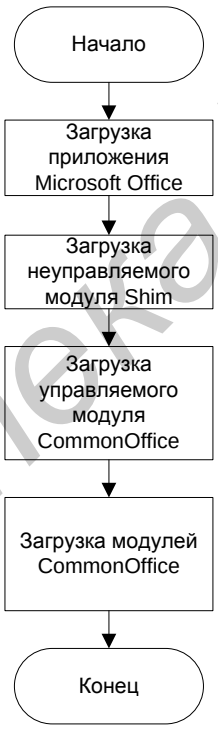

Рис. 1 – Блок-схема общего механизма инициализации CommonOffice в приложении Microsoft Office.

Таким образом, CommonOffice предоставляет прикладной интерфейс для программирования модулей расширения Microsoft Office приложений и берет на себя ответственность за загрузку, управление временем жизни и взаимодействие модуля как с хост приложением, так и с другими модулями. CommonOffice эффективно размещает модуля в памяти, создает производительную инфраструктуру для их взаимодействия, избегая дублирование кода и данных путем экспортирования общих сервисов (аутентификация, протоколирование и прочее) в отдельное адресное пространство (вспомогательный процесс).

Результатом разработки будет являться программное средство, которое можно внедрять коллективам программистов в качестве инструментария для дальнейшей разработки модулей расширения Microsoft Office приложений.

Список использованных источников:

1. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер // Спб.:Питер, 2013. – 896 с.
2. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес // Спб.:Питер, 2013. – 368 с.